Interactions 3D coopératives en environnements virtuels avec OpenMASK pour l'exploitation d'objets techniques

Thierry Duval, Christian Le Tenier

Interactions 3D coopératives en environnements virtuels avec OpenMASK pour l'exploitation d'objets techniques

Thierry Duval, Christian Le Tenier

IRISA projet SIAMES
Campus de Beaulieu, 35042 Rennes Cedex, FRANCE
02 99 84 72 56 – 02 99 84 25 26
E-mail : {Thierry.Duval,Christian.Le_Tenier}@irisa.fr

Résumé : Nous présentons ici les mécanismes que nous implémentons au dessus de la plate-forme OpenMASK dans le but de faciliter l'interaction dans les mondes virtuels 3D, en particulier lorsqu'ils sont peuplés d'objets à forte composante technique, c'est-à-dire imposant de fortes contraintes lors des interactions 3D. Nous présentons les mécanismes génériques utilisés pour permettre d'interagir avec des objets ainsi que pour obtenir des informations permettant de visualiser le plus explicitement possible les possibilités d'interactions offertes par les objets ainsi que les interactions effectivement en cours. Nous terminons par la façon de rendre explicite les interactions des autres utilisateurs dans le cas des univers 3D coopératifs.
Mots clés : Réalité virtuelle coopérative, interactions 3D, prise de conscience de l'interaction, prise de conscience de la coopération .
Abstract : In this paper we present the mechanisms we implement within the OpenMASK framework with the aim to make more easy the 3D interactions within virtual worlds. We are interested in technical objects with a lot of constraints during the interactions. We talk about the generic mechanisms used to interact with virtual objects and to make the users aware of the interaction services offered by the interactive objects, before and during the interactions. Finally we explain how we take into account the need to make the users aware of other users interactions within collaborative virtual universes.
Keywords : Collaborative Virtual Reality, 3D Interactions, Interaction Awareness, Collaboration Awareness.

1. Introduction

Il s'agit ici d'autoriser plusieurs participants à interagir au sein d'un même environnement 3D, avec les éléments de cet univers. Contrairement à des univers massivement coopératifs comme ceux d'"Avatars" [1], dans lesquels beaucoup de personnes peuvent se rencontrer mais ne peuvent pas produire grand chose dans l'univers virtuel, nous ciblons plutôt des univers partagés par un petit nombre de personnes qui vont pouvoir coopérer finement sur des objets de l'univers dans le but de produire un résultat tangible. Ces objets partagés, de grande technicité, peuvent être issus de milieux industriels variés (aéronautique, automobile, énergétique) et peuvent nous imposer des contraintes métiers exigeantes pour les interactions. Pour une interaction efficace les participants doivent avoir une perception explicite des possibilités d'interactions offertes d'une part par les objets de l'univers, comme dans [9], et d'autre part par les outils d'interaction mis à leur disposition. Enfin, comme les participants d'une session coopérative de travail peuvent être situés sur des sites distants, ils doivent avoir la possibilité de percevoir de façon explicite, en plus de leurs propres actions, celles des autres utilisateurs, comme dans [4][8] pour rendre la coopération plus efficace et plus facile à comprendre. Cette explicitation des actions des autres nous semble particulièrement importante comme le montrent de nombreux travaux du domaine [5][6][7]. C'est pourquoi nous souhaitons ici renforcer encore la prise de conscience des actions de l'utilisateur et celles des autres au cours d'une session de travail coopérative, ce que nous allons présenter ici dans nos travaux utilisant la plate-forme Open-MASK [10].

2. Une approche générique de l'interaction

2.1. Définition d'un protocole de dialogue

Notre approche consiste à définir des protocoles d'interaction entre d'une part des objets susceptibles d'être manipulés, et d'autre part des outils susceptibles de manipuler ces objets. Chacun de ces protocoles d'interaction peut se décomposer en règles d'interactions et en un protocole de communication. Ce dernier définit un ensemble de messages qui seront échangés lors du processus d'interaction entre l'outil d'interaction et l'objet soumis à l'interaction. Ces protocoles de communication sont porteurs d'une sémantique d'action plus ou moins riche. Ainsi, un premier protocole ne se composera que des quelques messages principaux nécessaires à une interaction (" prise de contrôle ",

" relâchement du contrôle "), alors qu'un protocole plus évolué s'enrichira de messages relatifs à la mise en évidence.
Par contre, les règles d'interaction vont définir le comportement qu'adopteront l'outil d'interaction et l'objet à contrôler suite à l'émission ou à la réception d'un message. Ainsi, plusieurs ensembles de règles d'interaction peuvent être proposés pour un même protocole de communication.
Un premier protocole d'interaction est implémenté sous OpenMASK, il impose quelques règles simples qui répondent aux questions que peut poser tout type d'interaction :

- Un outil d'interaction peut-il contrôler plusieurs objets de simulation simultanément ?
- Un objet de simulation peut-il être contrôlé par plusieurs outils d'interaction en même temps ?
- Un outil d'interaction accepte-t-il de terminer une interaction lorsque l'objet qu'il contrôle le lui demande ?
- Un objet de simulation accepte-t-il que l'outil d'interaction qui le contrôle puisse interrompre l'interaction en cours ?

Les réponses à ces questions dépendront du comportement que l'on voudra que les outils d'interaction et objets de simulation adoptent lors d'une interaction, et les règles d'interactions sont la traduction de ces réponses en une logique programmée.
Les diagrammes de séquences des figures 1, 2 et 3 donnent une description du protocole d'interaction cité précédemment, au travers de trois cas de prise de contrôle d'un objet de simulation par un outil d'interaction.
Les messages du protocole de communication transitent dans les deux sens via un contrôleur. La réception de ces messages provoque de part et d'autre des réactions qui sont définies par les règles d'interactions.
Les deux objets de simulation sont yellow cube et blue cube, les deux outils d'interactions sont virtual ray et Xinteractor.

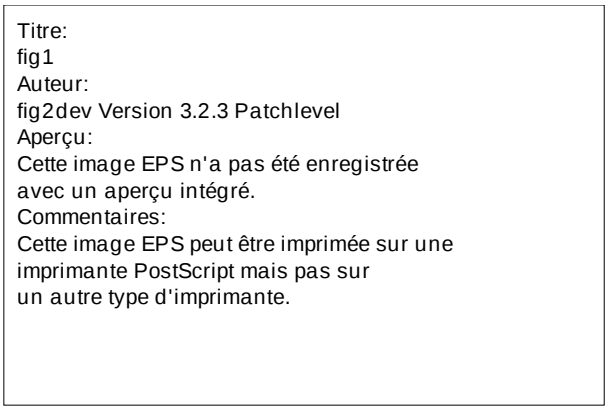

Fig. 1 : pas encore d'interaction en cours

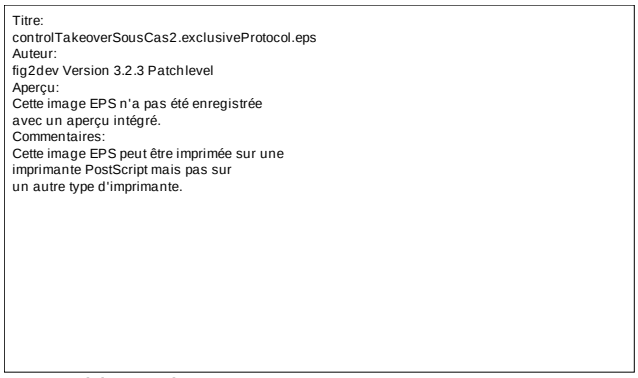

Fig. 2: blue cube est déjà sous le contrôle d'un autre outil d'interaction.

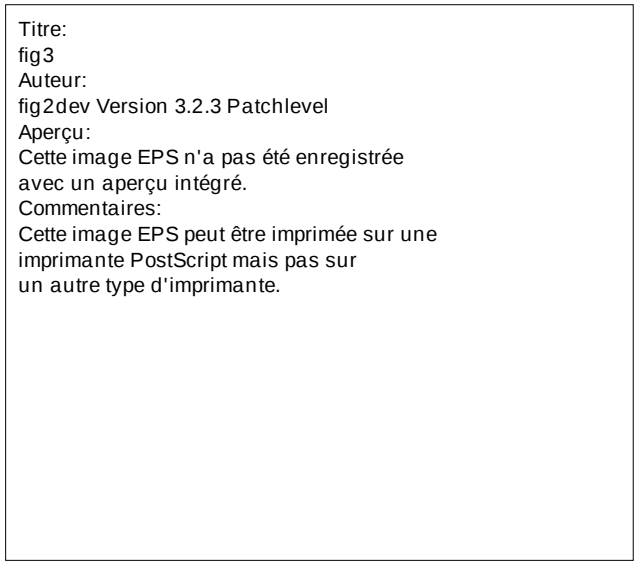

Fig. 3: virtual ray est déjà en interaction avec un autre objet de simulation.

2.2. Apprentissage de l'interaction

D'une simulation 3D à une autre, on peut vouloir retrouver les mêmes objets de simulation, parce qu'ils offrent des comportements qui nous intéressent, mais on peut également vouloir interagir avec eux différemment. Cependant, on ne désire pas avoir autant de versions d'un objet de simulation qu'il peut y avoir de protocoles d'interaction.
Ainsi, il nous faut dissocier le protocole d'interaction des objets qui l'utilisent. Ceci revient à utiliser des mécanismes qui permettront à un objet de simulation d'apprendre un protocole d'interaction, c'est-à-dire de rendre interactif un objet qui ne l'était pas. Un objet de simulation peut donc être conçu indépendamment des protocoles d'interaction qu'il apprendra à utiliser ultérieurement.

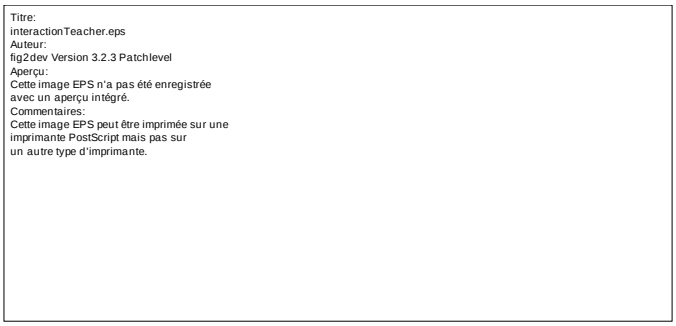

Fig. 4 : apprentissage d'un protocole d'interaction

Dans le cadre d'OpenMASK nous utilisons les possibilités qu'offrent les templates du C++ dont la figure 4 fournit un exemple d'application. Cette figure nous fournit une représentation de la manière dont un objet de simulation (brut ou adapté) peut apprendre un protocole d'interaction et devenir ainsi capable de répondre à une interaction. La partie gauche de la figure 4 donne la représentation d'un objet adapté dans le cadre d'OpenMASK. Ainsi, on voit comment un objet possédant des entrées, sorties et paramètres de contrôle peut être encapsulé pour apprendre un protocole d'interaction. La partie droite de la figure 4 est une représentation UML qui décrit plus précisément le mécanisme utilisé par la classe InteractionTeacher pour enseigner à la classe simulatedObject un protocole d'interaction.
En poussant le raisonnement, nous séparons notre monde d'interaction entre les objets qui permettent d'interagir et ceux qui subiront une interaction. Nous appellerons les premiers des outils d'interactions et les seconds des objets manipulables.
Ces outils seront des métaphores d'interaction qui apprendront comment interagir grâce à des interacteurs et les seconds sont de simples objets de simulation qui auront appris à réagir à une interaction grâce aux adaptateurs.
En effet, les interacteurs et adaptateurs sont les mécanismes qui nous permettent de créer des interactions et de mettre en place notre protocole d'interaction.

2.3. Assistance à la création d'objets manipulables : les adaptateurs

Une fois un tel protocole défini, nous offrons des classes génériques permettant d'apprendre ce protocole à des objets conçus par des programmeurs qui ignoraient ces possibilités d'interactions et la forme que de telles interactions pourraient prendre. Nous appelons ces classes des adaptateurs, ce sont des classes génériques qui permettent de prendre le contrôle, lors d'interactions, de certains paramètres d'états des objets adaptés. La détermination du type et du nom de ces paramètres se fait en paramétrant de façon adéquate ces adaptateurs.

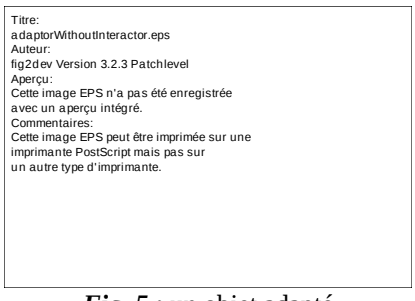

Fig. 5 : un objet adapté

La figure 5 nous fournit une représentation d'un objet adapté et nous donne une idée du fonctionnement d'un adaptateur. En effet, en étant encapsulée par la classe Adapter, la classe technicalObjet apprend un protocole d'interaction et se voit rajouter de nouvelles entrées, auxquelles un interacteur connectera dynamiquement ses sorties d'interaction, et dont les valeurs surchargeront celles de ses paramètres d'état (voir figure 6). Donc, sachant que les valeurs de ces paramètres d'état sont répercutés vers les sorties de l'objet adapté, on peut voir que lors d'une interaction le comportement propre à l'objet adapté est modifié (le plus souvent partiellement) par l'action d'un interacteur .

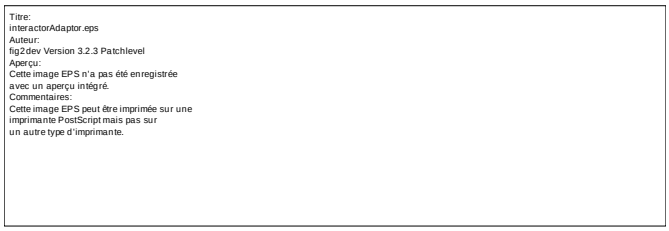

Fig. 6 : une interaction en cours

2.4. Création d'un objet de simulation interactif

Tout objet de simulation est construit suivant certaines règles, c'est-à-dire suivant un Framework, qui lui permettent d'être utilisé dans le cadre d'une application OpenMASK. Ainsi, toute nouvelle classe d'objet de simulation doit hériter de la classe PsSimulatedObject et implémenter, entre autres, une méthode compute(). Cependant, pour faciliter la création d'un objet interactif, nous proposons une surcouche à ce Framework.
En effet, tel que le montre la figure 7 pour la classe d'objet de simulation MotionLess, nous nous proposons d'hériter de la classe InteractiveObject qui redéfinit la méthode compute() en trois méthodes distinctes. Ces méthodes permettront de manipuler plus finement un objet adapté lors d'une interaction et notamment ses paramètres d'état (ou paramètres de contrôle) .
Cependant, il est bien entendu que nous proposons une interface qui permet de préparer un objet de simulation à devenir interactif, mais un programmeur averti pourra redéfinir lui-même la méthode compute() de son objet de simulation sans utiliser cette interface.
Pour information, dans la figure 7, la classe MotionLess, en héritant de la classe PsvMechanismPartner, acquiert la propriété d'être visualisable à l'intérieur d'une application OpenMASK (ce que nous appelons une session de simulation ou encore tout simplement une simulation).

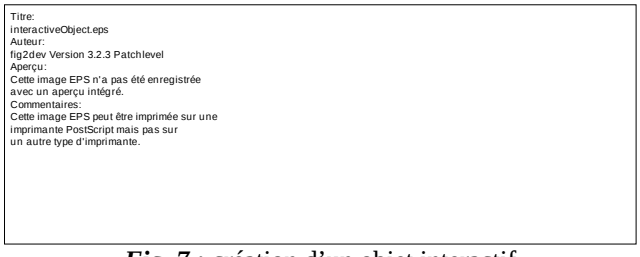

Fig. 7 : création d'un objet interactif

2.5. Assistance à la création d'outils d'interaction : les interacteurs

Symétriquement, nous fournissons des outils d'interaction qui connaissent la partie complémentaire du protocole de dialogue avec les objets manipulables. Ici encore, nous cherchons à offrir un maximum de généricité en encapsulant dans des classes génériques des classes de base qui s'appuient sur des pilotes de périphériques, afin de s'abstraire au maximum des contraintes matérielles. Nous appellerons ces classes des interacteurs.
La figure 8 nous fournit un exemple d'interacteur dont l'utilisation est identique à celle d'un adaptateur. Cependant, l'objet qui sera encapsulé par l'interacteur n'est plus un simple objet de simulation, mais un objet dédié dès sa conception à devenir un outil d'interaction. Il est conçu autour d'une métaphore d'interaction telle que celle du rayon virtuel ou celle de la main (utilisation de gants de données) qui peuvent se décliner en plusieurs variantes. Pourtant, elles seront toujours conçues sans se préoccuper du protocole d'interaction qui sera utilisé.

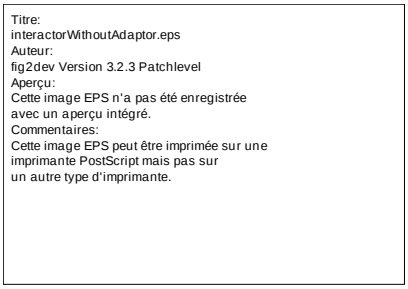

Fig. 8 : un interacteur

2.6. Extensions des protocoles, des adaptateurs et des interacteurs

De nombreuses possibilités d'interactions s'offrent alors :

- il est possible de réutiliser les objets à manipuler et les supports d'outils pour les encapsuler respectivement dans de nouveaux adaptateurs et de nouveaux interacteurs qui définissent de nouveaux protocoles de dialogue pour l'interaction,
- il est aussi possible de créer de nouveaux adaptateurs par héritage pour changer le comportement associé aux objets adaptés tout en gardant la connaissance du protocole d'interaction,
- symétriquement, on peut créer de nouveaux interacteurs par héritage pour changer le comportement associé aux outils d'interaction, toujours en conservant la connaissance du protocole tout en changeant le mode de contrôle des objets manipulés.

3. Mise en évidence de l'interaction

3.1. Informations véhiculées lors des échanges

Les protocoles associés à nos adaptateurs et interacteurs incluent des échanges d'informations, déclenchés par exemple lors de désignations des objets manipulables par les outils d'interaction, qui permettent de transmettre aux manipulateurs et au dispositif de visualisation des informations quant au type des paramètres d'état qu'ils mettent ainsi à la disposition de l'utilisateur. Ainsi que le montre la figure 9, avant toute interaction on peut obtenir des informations sur les objets en présence. Ici, les informations fournies ont trait à la manière dont les cônes peuvent être manipulés lors d'une interaction. Ainsi le cône du centre pourra être manipulé selon 3 Degrés De Liberté (DDL) en translation et 3 DDL en rotation (image gauche), le cône de gauche sera manipulable selon 3 DDL en translation (image centrale) et le cône de droite selon 3 DDL en rotation (image droite). L'utilisateur indique dans un fichier de configuration s'il veut utiliser ou non ce mode d'informations pour tout ou partie des objets de la simulation.
De plus, on voit que la forme géométrique qui fournit des informations quant à son mode de manipulations est mise légèrement en surbrillance. Cette surbrillance indique à l'utilisateur que cette géométrie est dans la sphère d'influence de son interacteur. En effet, il est quelquefois malaisé dans une Réalité Virtuelle d'avoir une réelle notion de la profondeur, ce qui peut générer des tâtonnements avant d'être en mesure de prendre le contrôle d'un objet. Ceci est particulièrement vrai lors de l'utilisation de gants de données en situation immersive. Ainsi, la surbrillance informera l'utilisateur quant à ses capacités d'interaction, ce qui lui permettra à coup sûr de prendre le contrôle de l'objet désiré.

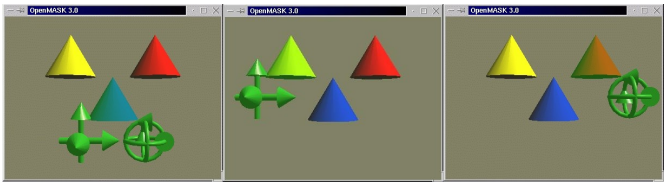

Fig. 9 : informations sur les paramètres d'états

De même, lors de d'interaction, un objet doit pouvoir signifier visuellement qu'il est sous le contrôle d'un interacteur. Ceci pour informer d'une part l'utilisateur qui manipule l'interacteur qu'il a le contrôle de l'objet, et d'autre part les autres participants de la simulation que cet objet est soumis à une interaction.
Plusieurs types de mise en évidence peuvent être utilisés, ceux qui sont actuellement disponibles avec notre plate-forme OpenMASK sont présentés figure 10. Le type de mise en évidence à utiliser est fourni à chaque objet d'une simulation dans le fichier de configuration de la simulation. Ainsi, dans une même simulation plusieurs types de mises en évidence peuvent être utilisés.
Sur la partie gauche de la figure 10 on peut voir un exemple de mise en évidence par duplication de la géométrie de l'objet sous contrôle. Cette mise en évidence a l'avantage d'être ergonomique mais peut être gourmande en termes de ressources si la géométrie initiale est complexe. Par contre, la partie droite présente une mise en évidence basée sur l'ajout d'une géométrie au format de la boîte englobante de l'objet sous contrôle. Comme on peut le voir, le principal inconvénient est l'encombrement visuel, qui peut amener quelquefois (pour certaines géométries) l'utilisateur à être inclus dans cette boîte englobante. Cependant, contrairement

à sa consœur, cette mise en évidence est peu gourmande en ressources et peut permettre de fournir sur l'interaction en cours des informations équivalentes aux informations fournies par les axes de la figure 9 via le choix de la géométrie à ajouter.

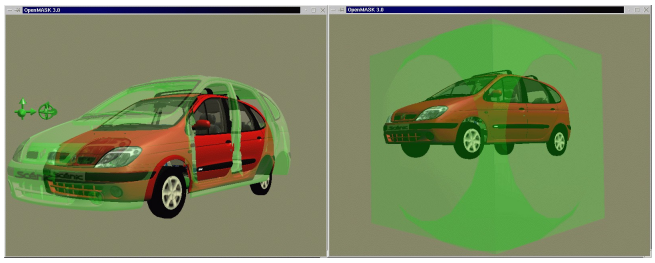

Fig. 10 : deux mises en évidence d'interaction

3.2. Mise en évidence des interactions, potentielles et effectives

De ce fait, il devient possible de mettre en évidence, par exemple comme illustré précédemment à l'aide d'une représentation graphique 3D, la nature de l'interaction qu'il est possible d'avoir avec les objets, pour une information avant interaction. De la même façon, nous répercutons également à l'utilisateur une information visuelle en cours d'interaction, d'une part pour l'informer de la validation de son choix d'interaction (parmi les différentes possibilités offertes), et d'autre part afin de le guider dans l'interaction. Ainsi, lors de la manipulation de maquettes numériques obtenues à partir de données CAO, il doit devenir possible de faire apparaître des contraintes de manipulation qui peuvent être liées à des liaisons mécaniques entre objets (liaisons pivot, rotule, glissière, . . .). Ces liaisons mécaniques sont d'ores et déjà implantées dans OpenMASK sous la forme d'adaptateurs spécialisés, comme on peut le voir dans la figure 11 où des interactions ont été effectuées avec des parties de véhicules fortement contraintes.

Fig. 11 : interactions avec contraintes techniques fortes

Par contre la mise en évidence de ces contraintes techniques n'est pas encore tout à fait terminée : actuellement la contrainte d'interaction due à une liaison pivot est visualisée de la même façon que la contrainte due à une liaison rotule, c'est-à-dire comme une rotation générique, par exemple à l'aide d'une sphère englobante semi-transparente.

4. Mise en évidence de la coopération

Enfin, en ce qui concerne la coopération, OpenMASK, tout comme GASP dont il dérive [2][3], nous permet de partager une session de travail en environnement virtuel sur différentes machines, au travers d'un réseau. Comme nous offrons une représentation visuelle des offres d'interaction, puis des interactions elles-mêmes, ces représentations graphiques se trouvent elles aussi distribuées et visualisées sur les différentes machines de la session coopérative. En représentant d'une façon différente les interactions locales (celles associées à un interacteur lié à la visualisation locale) et les interactions distantes (celles liées aux interacteurs associés aux autres visualisations, sur les autres machines), nous permettons alors à chaque utilisateur de percevoir aussi les actions des autres utilisateurs avec qui il peut alors plus facilement coopérer.
Les figures 12 à 16 vont illustrer un mode de représentation des mises en évidence de la coopération basées sur des codes de couleur. Il s'agit ici d'une coopération entre 2 utilisateurs situés sur 2 stations de travail nommées gwinver et asterix. L'écran de gauche représentera la vue de l'utilisateur de la station gwinver et l'écran de droite représentera celle de l'utilisateur de la station asterix.

4.1. Mise en évidence des interactions des autres

Étudions tout d'abord le cas des objets manipulables en mode exclusif, c'est-à-dire par un seul utilisateur à la fois.
Pour faire prendre conscience à un utilisateur qu'il contrôle un tel objet, nous avons ici choisi de donner une couleur verte semi-transparente au mode de mise en évidence de l'interaction, lorsque l'on voit l'objet dans la visualisation associée à l'utilisateur, et de donner une couleur rouge lorsque l'on voit l'objet dans une visualisation associée à un autre utilisateur. Pour lever toute ambiguïté nous avons également décidé de placer le nom de l'utilisateur près de l'objet manipulé. On peut voir figure 12 le résultat d'une prise de contrôle du cône bleu par l'utilisateur de gwinver, vu par chacun des utilisateurs de gwinver et d'asterix.

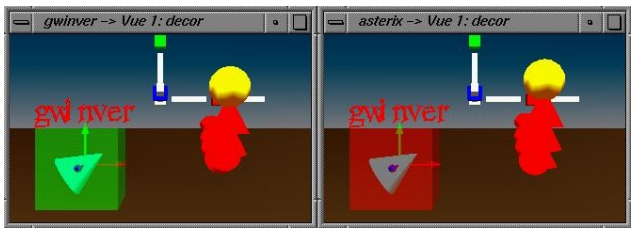

Fig. 12 : l'utilisateur de gwinver a le contrôle du cône bleu

La figure 13 montre quant à elle ce que voient ces 2 utilisateurs lorsque l'utilisateur d'asterix a pris à son tour le contrôle de l'objet interactif.

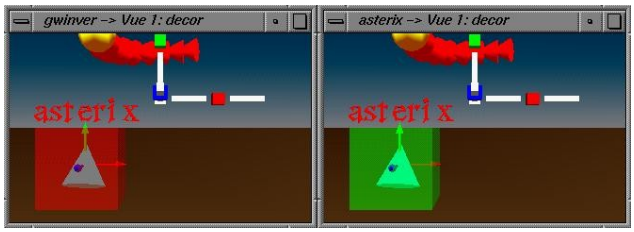

Fig. 13 : l'utilisateur d'asterix a le contrôle du cône bleu

Bien entendu, chaque utilisateur peut interagir avec un objet de l'univers partagé, ce qui permet aux utilisateurs de coopérer comme illustré figure 14 où l'utilisateur d'asterix positionne le cône pendant que l'utilisateur oriente ce même cône via un placement de la sphère vers laquelle ce cône est sensé pointer.

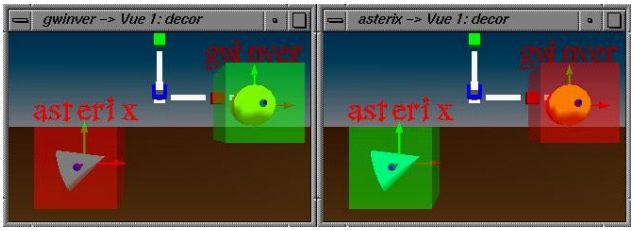

Fig. 14 : chaque utilisateur contrôle un objet

4.2. Cas des interactions simultanées sur un objet

Nous avons également développé des adaptateurs permettant à plusieurs utilisateurs d'agir simultanément sur un même objet, en adaptant la mise en évidence des interactions sur ces objets. Comme le nombre d'utilisateurs simultanés sur un même objet peut ne pas être limité, nous n'affichons pas systématiquement les noms des utilisateurs en interaction pour ne les afficher que sur requête explicite d'un utilisateur, et ce de façon locale pour ne pas induire de surcharge visuelle chez les autres utilisateurs.
Nous jouons encore ici sur des codes de couleurs : un objet partageable en interaction se verra associé à une mise en évidence bleue si l'utilisateur est l'un de ceux qui contrôlent l'objet, et la mise en évidence sera blanche sinon. Pour donner une information minimale, c'est le nombre d'utilisateurs en interaction avec l'objet qui sera systématiquement visualisé.
La figure 15 illustre un cas d'interaction avec un tel objet : l'utilisateur de gwinver a pris le contrôle d'un objet partageable, il est informé qu'il est le seul utilisateur en interaction (à cause du nombre et de la couleur), il ne demande donc pas à voir la liste des utilisateurs en interaction avec l'objet. De l'autre côté, l'utilisateur d'asterix est informé qu'un autre utilisateur est en interaction avec cet objet : il demande donc à voir quel est cet utilisateur.

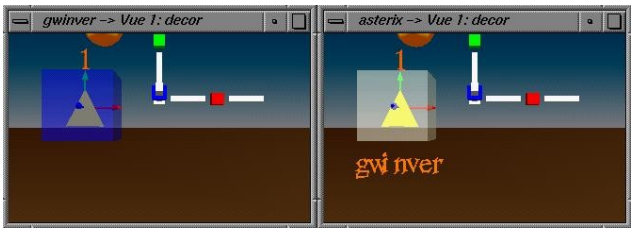

Fig. 15 : objet fortement partageable contrôlé par un seul utilisateur

On peut voir figure 16 que l'utilisateur d'asterix a lui aussi choisi d'interagir avec le cône jaune, et qu'il a demandé à cet objet de faire disparaître les noms des utilisateurs. De l'autre côté, l'utilisateur de gwinver a perçu qu'un autre utilisateur était en train de partager son interaction avec le cône jaune, et il a demandé à cet objet quels étaient ses autres utilisateurs.

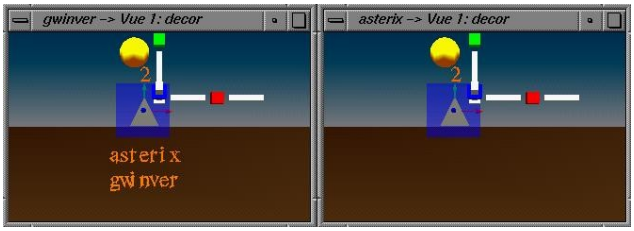

Fig. 16 : objet fortement partageable contrôlé par deux utilisateurs simultanément

5. Conclusion

Nous avons donc présenté les mécanismes que nous utilisons pour permettre l'interaction avec des objets placés dans des mondes virtuels qui peuvent être partagés par plusieurs utilisateurs, grâce à la plate-forme OpenMASK. Nous avons ensuite montré comment nous ajoutions les informations permettant d'expliciter visuellement le potentiel interactif des objets de l'univers ainsi que les interactions effectives avec ces objets, avant de parler des interactions contraintes pour des objets fortement techniques. Ces univers virtuels étant partageables par plusieurs utilisateurs, la représentation des interactions d'un utilisateur peut alors être facilement propagée aux autres utilisateurs afin de leur fournir une perception du travail des autres, à condition d'offrir une visualisation différente des mises en évidence pour l'utilisateur de celles offertes aux autres. Enfin, nous avons abordé le cas de la mise en évidence des interactions simultanées de plusieurs utilisateurs sur un même objet.

6. Bibliographie

[1] B. Damer, S. Gold, J. de Bruin & D.-J. de Bruin. Conferences and Trade Shows in Inhabited Virtual Worlds: A Case Study of Avatars98 & 99. In Proceedings of the Second International Conference on Virtual Worlds (VW'2000), Springer LNCS/AI, Paris, France, 1-11, 2000.
[2] T. Duval, D. Margery. Using GASP for Collaborative Interactions within 3D VirtualWorlds. In Proceedings of the Second International Conference on Virtual Worlds (VW'2000), Springer LNCS/AI 1834, Paris, France, 65-76, 2000.
[3] T. Duval, D. Margery. Buiding Objects and Interactors for Collaborative Interactions with GASP. In Proceedings of the Third International Conference on Collaborative Virtual Environments (CVE'2000), ACM, San Francisco, USA, 129-138, 2000.

[4] M. Fraser, S. Benford, J. Hindmarsh, C. Heath. Supporting Awareness and Interaction through Collaborative Virtual Interfaces. In Proceedings of UIST'99, ACM, Asheville, USA, 27-36, 1999.
[5] M. Fraser, T. Glover, I. Vaghi, S. Benford, C. Greenhalgh, J. Hindmarsh, C. Heath. Revealing the Realities of Collaborative Virtual Reality. In Proceedings of the Third International Conference on Collaborative Virtual Environments (CVE'2000), ACM, San Francisco, USA, 29-37, 2000.
[6] C. Gutwin, S. Greenberg. The Effects of Workspace Awareness Support on the Usability of Real-Time Distributed Groupware. In ACM Transactions on Computer-Human Interaction 6(3), 243-281, 1999.
[7] C. Gutwin, S. Greenberg. Design for Individuals, Design for Groups: Tradeoffs Between Power and Workspace Awareness. In Proceedings of CSCW'98, ACM, Seattle, USA, 207-216, 1998.
[8] J. Hindmarsh, M. Fraser, C. Heath, S. Benford, C. Greenhalgh. Fragmented Interaction: Establishing mutual orientation in virtual environments. In Proceedings of CSCW'98, ACM, Seattle, USA, 217-226, 1998.
[9] M. Kallman, D. Thalmann (1999), Direct 3D Interaction with Smart Objects, in Proceedings of VRST'99, ACM, Lisbon, Portugal, 124-130, 1999.
[10] D. Margery, B. Arnaldi, A. Chauffaut, S. Donikian, T. Duval. OpenMASK: Multi-Threaded or Modular Animation and Simulation Kernel or Kit: a General Introduction. In proceedings of VRIC'2002, Laval, France, 2002.